УДК 510.63+519.72

А. Г. Малев, В. Ю. Попов*

АЛГОРИТМ ПРЕДСКАЗЫВАЮЩЕГО КОДИРОВАНИЯ ПРИ ПОМОЩИ ПОСТРОЕНИЯ СДНФ

Сжатие данных — это форма кодирования битового потока, позволяющая удалением избыточности уменьшить его размеры. Сжатие сокращает объем пространства, требуемого для хранения файлов, и количество времени, необходимого для передачи информации по каналу фиксированной пропускной способности. Удаляя из данных избыточность, сжатие способствует успешному применению шифрования, затрудняя криптоанализ статистическим методом. С другой стороны, алгоритмы сжатия данных могут быть использованы при криптоанализе для выявления избыточности и статистических закономерностей. В последние годы сжатие активно используется для ускорения обработки информации: обработка сжатой информации в оперативной памяти выгоднее обработки исходного массива информации, размещенного на внешнем носителе. Таким образом, сжатие является одним из важнейших методов обработки информации.

Согласно классической теореме Шеннона [1], не существует абсолютно сжимающего преобразования, т. е. преобразования, позволяющего уменьшить любой битовый поток. Поэтому алгоритмы сжатия обычно разрабатываются не для произвольных массивов информации, а для конкретных классов, принадлежность к которым может быть установлена по некоторым легко проверяемым внешним признакам: тексты, исполняемые файлы, графика, звук, видео и т. д. На сегодняшний день существует устоявшаяся классификация массивов информации по типу их сжатия. Одним из важнейших классов являются однобитные изображения: деловая графика, факсимильные и сканированные изображения текстов и т. д.

Алгоритмы сжатия однобитных изображений естественным образом делятся на два класса: алгоритмы, учитывающие двумерную структуру изображения, и алгоритмы одномерного сжатия. Среди алгоритмов первого типа стандартом де-факто является JBIG, полное описание которого можно найти по ссылке [2]. А среди алгоритмов второго типа — CCITT Group 4 (Consultative Committee for International Telegraphy and Telephony). В открытом доступе можно найти лишь полное описание алгоритма CCITT Group 3

*Работа выполнена при частичной поддержке гранта Президента Российской Федерации для государственной поддержки молодых российских ученых МД-8770.2006.9.

(см. [3]). Однако оба алгоритма основаны на одной и той же идее сжатия, где последовательности подряд идущих черных и белых точек заменяются числом, равным их количеству, а потом применяется алгоритм Хаффмана с фиксированной таблицей. Различаются они только использованной таблицей, которая для алгоритма CCITT Group 3 общедоступна, а для CCITT Group 4 — нет. Тем не менее доступ к программной реализации CCITT Group 4 можно получить, используя ACDSee 8.0 [4].

Цель данной статьи — рассмотрение способа дополнительной обработки массивов информации, формируемых алгоритмом CCITT Group 4, для повышения степени сжатия.

1. Описание алгоритма сжатия

Любое черно-белое изображение можно рассматривать как однобитовую матрицу. Используемые нами методы не опираются на двумерную структуру входа. Поэтому мы будем работать не с матрицей, а с последовательностью бит, полученной путем конкатенации строк этой матрицы.

Исходной информацией для алгоритма сжатия является последовательность бит изображения и натуральное число ℓ — количество бит предсказания. На выходе алгоритм формирует сжатую битовую последовательность и дополнительную информацию для восстановления: СДНФ и первые ℓ бит исходной последовательности.

Алгоритм заключается в последовательном выполнении следующих четырех этапов:

1) CCITT Group 4;
2) построение СДНФ для предсказывающего кодирования;
3) формирование ошибки предсказания;
4) применение к ошибке предсказания алгоритма арифметического сжатия (см. [5]).

Первый этап заключается в применении уже упомянутого выше алгоритма CCITT Group 4 к исходной последовательности бит. Следует отметить, что при исследовании использовалась уже готовая реализация алгоритма, поставляемая вместе с системой ACDSee 8.0. Что касается четвертого этапа, то подробное описание метода арифметического сжатия можно найти в [5], а исходные коды — по ссылке [6]. При исследовании было замечено, что особенности реализации арифметического сжатия не сильно влияют на результат. Поэтому можно использовать и любую другую реализацию. Таким образом, в пояснении нуждаются лишь второй и третий этапы.

1.1. Построение СДНФ *для предсказывающего кодирования*

Пусть для любого x выражение x^α, где $\alpha \in \{0,1\}$, обозначает x при $\alpha = 1$ и $\neg x$ при $\alpha = 0$. Произвольную последовательность

$$w_1, w_2, \ldots, w_m,$$

где $w_i \in \{0,1\}$, мы будем в дальнейшем обозначать w или $w[1,m]$. При этом $w[i,j]$ для $1 \le i \le j \le m$ есть сегмент $w_i, \ldots, w_j$ последовательности w.

СДНФ $F(x_1, \ldots, x_\ell)$ строится по последовательности бит v, формируемой алгоритмом CCITT Group 4, последовательным добавлением дизъюнктов. Для каждой последовательности $w[1,\ell]$ найдем все последовательности $v[i, i+\ell]$ такие, что $w[1,\ell] = v[i, i+\ell-1]$. Если среди последовательностей $v[i, i+\ell]$ не менее половины удовлетворяют условию $v_{i+\ell} = 1$, то в СДНФ добавляется дизъюнкт

$$x_1^{w_1} \wedge x_2^{w_2} \wedge \ldots x_\ell^{w_\ell}.$$

1.2. Формирование ошибки предсказания

СДНФ $F(x_1, \ldots, x_\ell)$ мы будем использовать в качестве оракула для предсказания значения бита $v_{i+\ell}$ по битам $v[i, i+\ell-1]$, полагая $v_{i+\ell}$ равным $F(v_i, \ldots, v_{i+\ell-1})$.

Естественно, реальное значение $v_{i+\ell}$ может не совпасть со значением, предсказанным оракулом. Поэтому для того, чтобы избежать потери информации, мы создаем последовательность, которая кодирует ошибку предсказания $u[1, N-\ell]$. Если значение $v_{i+\ell}$ предсказано верно, мы полагаем u_i равным 0, в противном случае $u_i = 1$. Формально это можно записать так:

$$u_i = F(v_i, \ldots, v_{i+\ell-1}) \oplus v_{i+\ell}. \tag{1}$$

Таким образом, мы определили преобразование, переводящее последовательность $v[1,N]$ в последовательность ошибок $u[1, N-\ell]$. Это преобразование в дальнейшем мы будем обозначать через B_ℓ.

2. Восстановление сжатой информации

Поскольку алгоритм CCITT Group 4 и алгоритм арифметического сжатия сжимают без потерь и не нуждаются в дополнительной информации для восстановления исходных данных, для обоснования корректности нашего алгоритма достаточно показать, что справедливо следующее утверждение.

Теорема 1. *Последовательность $v[1,N]$ восстанавливается однозначно по* СДНФ *$F(x_1, \ldots, x_\ell)$, последовательности ошибок $u[1, N-\ell]$ и сегменту $v[1,\ell]$.*

Доказательство. По СДНФ $F(x_1, \ldots, x_\ell)$ и последовательности ошибок $u[1, N-\ell]$ и $v[1, \ell]$ построим последовательность $w[1, N]$ следующим образом:

$$w_p = \begin{cases} F(v_{p-\ell}, \ldots, v_{p-1}) \oplus u_p, & p > \ell, \\ v_p, & p \le \ell. \end{cases} \tag{2}$$

Покажем, что $v_p = w_p$, $1 \le p \le N$. В самом деле, при $p \le \ell$ равенство $w_p = v_p$ имеем по определению. А при $p > \ell$, согласно построению последовательности $w[1, N]$ и равенству (1),

$$w_p = F_p(v_{p-\ell}, \ldots, v_{p-1}) \oplus u_p = F_p(v_{p-\ell}, \ldots, v_{p-1}) \oplus F_p(v_{p-\ell}, \ldots, v_{p-1}) \oplus v_p = v_p,$$

что и требовалось.

3. Обоснование сжимающих свойств преобразования B_ℓ

Заметим, что, создавая последовательность ошибок $u[1, N-\ell]$, преобразование B_ℓ уменьшает последовательность v, получаемую от алгоритма CCITT Group 4, лишь на ℓ бит. При этом для восстановления информации мы должны хранить $v[1, \ell]$ и СДНФ, т. е. после второго и третьего этапов работы алгоритма количество бит по сравнению с результатом первого этапа не уменьшится, а даже возрастет. Поэтому мы нуждаемся в специальном обосновании целесообразности использования второго и третьего этапов.

Для того чтобы оценить качество сжатия конечной последовательности бит $v[1, n+m]$, содержащей n бит, равных 1, и m бит, равных 0, введем следующий аналог энтропии по Шеннону:

$$H_v = -n \cdot \log_2 \left(\frac{n}{n+m} \right) - m \cdot \log_2 \left(\frac{m}{n+m} \right).$$

Справедливы соотношения $H_v \ge 0$ и $H_v \le n+m$, причем равенство в последнем имеет место только при $n = m$. Согласно теореме Шеннона, величина $\lceil H_v \rceil$ показывает минимальное среднее число бит, необходимых для представления исходной последовательности бит v. То есть какой бы энтропийный метод сжатия не использовался, после его применения последовательность будет содержать в среднем не менее $\lceil H_v \rceil$ бит. Заметим, что существуют алгоритмы сжатия, которые достигают этого минимума. Например, хорошо известный метод арифметического сжатия.

Второй и третий этапы алгоритма предназначены не для непосредственного сжатия данных, а для модификации величины H_v для улучшения работы арифметического сжатия на четвертом этапе. Обоснование этого содержится в следующем утверждении.

Теорема 2. *Пусть $v[1,N]$ — произвольная последовательность бит и для некоторого натурального числа ℓ имеет место равенство $u[1,N-\ell] = B_\ell(v)$. Тогда $H_u \le H_{v[\ell+1,N]}$.*

Доказательство. Занумеруем всевозможные последовательности бит $x[1,\ell]$ числами от 1 до 2^ℓ. Через n и m обозначим число единиц и нулей последовательности $v[\ell+1,N]$, а через s и t — число единиц и нулей последовательности u. Для каждой последовательности $x[1,\ell]$ с номером k определим

$$\begin{aligned} n_k &= \sum_{i=1}^{N-\ell} v_{i+\ell} \wedge (v[i,i+\ell-1] = x[1,\ell]), \\ m_k &= \sum_{i=1}^{N-\ell} \neg v_{i+\ell} \wedge (v[i,i+\ell-1] = x[1,\ell]), \\ s_k &= \sum_{i=1}^{N-\ell} u_i \wedge (v[i,i+\ell-1] = x[1,\ell]), \\ t_k &= \sum_{i=1}^{N-\ell} \neg u_i \wedge (v[i,i+\ell-1] = x[1,\ell]). \end{aligned}$$

Непосредственно из определения следует, что

$$\sum_{k=1}^{2^\ell} n_k = n, \quad \sum_{k=1}^{2^\ell} m_k = m, \quad \sum_{k=1}^{2^\ell} s_k = s, \quad \sum_{k=1}^{2^\ell} t_k = t.$$

Заметим, что если $n_k \ge m_k$, то СДНФ преобразования B_ℓ на последовательности бит с номером k предсказывает значение, равное 1, и тогда $t_k = n_k \ge m_k$. А если $n_k < m_k$, то значение СДНФ на последовательности бит с номером k равно 0, и поэтому $t_k = m_k > n_k$. Таким образом, для любого значения k имеют место неравенства $t_k \ge n_k$ и $t_k \ge m_k$. Следовательно, $t \ge m$ и $t \ge n$. Поскольку $n + m = t + s = N - \ell$, имеют место неравенства $s \le n$, $s \le m$, $t \ge \frac{N-\ell}{2}$ и $s \le \frac{N-\ell}{2}$. В силу монотонного возрастания функции $-x \cdot \log_2(x)$ при $0 \le x \le \frac{1}{2}$ и убывания при $\frac{1}{2} \le x \le 1$, получим:

$$\begin{aligned} H_u &= -s \cdot \log_2\left(\frac{s}{N-\ell}\right) - t \cdot \log_2\left(\frac{t}{N-\ell}\right) \le \\ &\le -n \cdot \log_2\left(\frac{n}{N-\ell}\right) - m \cdot \log_2\left(\frac{m}{N-\ell}\right) = H_{v[\ell+1,N]}, \end{aligned}$$

что и требовалось доказать.

4. Результаты эксперимента

Исследование всех алгоритмов проводилось на стандартном наборе черно-белых изображений. Этот набор можно найти по ссылке [7]. При исследовании алгоритма внимание уделялось только степени сжатия, т. е. отношению исходного размера файла к размеру архива.

В таблице приведены результаты исследования алгоритма с СДНФ от 12 переменных (обозначенного нами A_{12}) в сравнении с другими алгоритмами.

Результаты сжатия тестовых изображений CCITT различными алгоритмами

Изображение	Исходный размер	CCITT Group 4	CCITT Group 4 + арифм. сжатие	A_{12}	WinRar v3.42 макс. сжатие
f01_200	505 286	16 770 3,32 %	16 480 3,26 %	15 942 3,16 %	24 768 4,90 %
f02_200	505 286	10 616 2,10 %	10 506 2,08 %	10 490 2,08 %	20 455 4,05 %
f03_200	505 286	25 742 5,10 %	24 624 4,87 %	23 634 4,68 %	36 884 7,30 %
f04_200	505 286	64 210 12,71 %	62 927 12,45 %	59 241 11,72 %	80 702 15,97 %
f05_200	505 286	29 436 5,83 %	28 599 5,66 %	27 479 5,44 %	43 219 8,56 %
f06_200	505 286	15 926 3,15 %	15 019 2,97 %	14 432 2,86 %	24 250 4,80 %
f07_200	505 286	66 584 13,18 %	66 020 13,07 %	65 116 12,89 %	89 208 17,65 %
f08_200	505 286	18 136 3,59 %	17 428 3,45 %	17 072 3,38 %	31 905 6,31 %
f10_200	504 638	142 242 28,19 %	127 075 25,18 %	96 898 19,20 %	73 360 14,54 %
Итого	4 546 926	389 662 8,57 % в 11,67 раз	368 678 8,11 % в 12,33 раз	330 304 7,26 % в 13,77 раз	424 751 9,34 % в 10,70 раз

Таблица в строках с первой по девятую содержит размеры в байтах файлов, полученных после применения соответствующих методов, и отношение к размеру исходного тестового изображения. Следует отметить, что алгоритм арифметического сжатия применялся не к исходным файлам, а к предвари-

тельно обработанным методом CCITT Group 4. В последней строке содержится сумма всех размеров исходных файлов S, а также по каждому методу: сумма размеров полученных файлов S_i, отношение этой суммы к исходной сумме в процентах — $S_i/S \cdot 100\,\%$ и средняя степень сжатия метода — величина, вычисляющаяся по формуле S/S_i.

Из таблицы видно, что наиболее эффективным из рассмотренных алгоритмов для сжатия черно-белых изображений является метод A_{12}. В среднем он сжимает в 13,77 раз, т. е. оставляет 7,26 % от исходного массива данных, в то время как алгоритм CCITT Group 4 обеспечивает сжатие в 11,67 раз, а алгоритм WinRar — в 10,7 раз.

Литература

1. Шеннон К. Э. Работы по теории информации. М.: Иностр. лит., 1966.

2. ISO/IEC Committee Draft 11544, Coded Representation of Picture and Audio Information // Progressive Bi-level Image Compression. 1992. Apr.

3. Ватолин Д. С. Алгоритмы сжатия изображений [Электрон. ресурс]. Режим доступа: http://algolist.manual.ru

4. [Электрон. ресурс]. Режим доступа: http://www.acdsystems.com

5. Howard P. G., Vitter J. S. Practical implementations of arithmetic coding // Image and Text Compression. Norwell, 1992. P. 85–112.

6. [Электрон. ресурс]. Режим доступа: http://www.compression.ru. Сайт посвящен алгоритмам сжатия данных.

7. [Электрон. ресурс]. Режим доступа: http://www.imagepower.com/compress/ccitt_images.htm. Набор тестовых черно-белых изображений CCITT Group 4.